FP-Redemption: Studying Browser Fingerprinting Adoption for the Sake of Web Security

Antonin Durey, Pierre Laperdrix, Walter Rudametkin, Romain Rouvoy

FP-Redemption: Studying Browser Fingerprinting Adoption for the Sake of Web Security

Antonin Durey[1], Pierre Laperdrix[2,1], Walter Rudametkin[1], and Romain Rouvoy[1,3]

[1] University of Lille / Inria
[2] CNRS
[3] IUF

Abstract. Browser fingerprinting has established itself as a stateless technique to identify users on the Web. In particular, it is a highly criticized technique to track users. However, we believe that this identification technique can serve more virtuous purposes, such as bot detection or multi-factor authentication. In this paper, we explore the adoption of browser fingerprinting for security-oriented purposes. More specifically, we study 4 types of web pages that require security mechanisms to process user data: sign-up, sign-in, basket and payment pages. We visited 1,485 pages on 446 domains and we identified the acquisition of browser fingerprints from 405 pages. By using an existing classification technique, we identified 169 distinct browser fingerprinting scripts included in these pages. By investigating the origins of the browser fingerprinting scripts, we identified 12 security-oriented organizations who collect browser fingerprints on sign-up, sign-in, and payment pages. Finally, we assess the effectiveness of browser fingerprinting against two potential attacks, namely stolen credentials and cookie hijacking. We observe browser fingerprinting being successfully used to enhance web security.

Keywords: browser fingerprinting · web security · cookies · multifactor authentication

1 Introduction

As web usage continues to grow, web security continues to be challenged. By exploiting vulnerabilities, such as credential leaks[4] from previous hacks or phishing [10] to obtain new credentials, hackers can log into websites and harvest users' data. Additionally, websites may be vulnerable to *cookie hijacking*, which consists of extracting cookies from a user session to access their accounts. These attacks can lead to data leaks, such as username, email, physical addresses or search history [28], but also to account hijacking. Given the growth of such attacks, *Multi-Factor Authentication* (MFA) is perceived as a reliable protection

[4] `https://haveibeenpwned.com/`

to increase web security. MFA consists in combining multiple security factors to check the identity of an authorized user. Such factors not only include user credentials, but also physical tokens, SMS codes, or dedicated mobile apps. These MFA solutions vary in price, level of security, and intrusiveness.

Among these alternatives, *browser fingerprinting* is a stateless identification technique [7, 17] that accesses attributes exposed by the browser and its environment to build a unique identifier. Over the years, studies have focused on exploring new attributes, and increasing the uniqueness of browser fingerprints. More specifically, several contributions studied browser fingerprinting for tracking purposes [1, 8, 32], while others focused on defending against it [31, 23, 16]. Some contributions proposed using browser fingerprinting as a new factor in MFA solutions [29, 2, 15], but have not evaluated the benefits to secure online websites. Moreover, given its identification potential, browser fingerprinting is useful for bot detection [14, 33], demonstrating its ability to detect undesired visitors.

In this paper, we investigate the adoption of browser fingerprinting to reinforce security on the web. Through our experiment, we intend to detect if fingerprinting is used to strengthen web security, and in which specific contexts this occurs. In particular, we target 4 types of web pages that store and process sensitive user information, namely sign-up, sign-in, basket and payment pages. Investigating these pages is a challenge as it is very hard to automate their exploration because of the sheer diversity of forms and payment processes. As such, we manually visited 1,485 pages from 446 websites belonging to 14 different categories with the aim of detecting fingerprinting scripts. By using an existing classification technique [6], we identified 169 fingerprinting scripts being used on all the secured types of pages we study, with 12 of them belonging to security-focused organizations. Finally, we study the resilience of websites adopting browser fingerprinting for security purposes by simulating two classes of attacks: *stolen credentials* and *cookie hijacking*.

The key contributions of this paper are:

- Evidence of the adoption of browser fingerprinting for security on 4 types of secure pages across various categories of websites, and a study of the browser fingerprints they extract.
- A dataset of 1,485 pages, obtained from 446 websites that include 169 distinct fingerprinting scripts;[5]
- A study of the resilience of websites that use fingerprinting to protect users from stolen credentials and hijacked cookies. In particular, we show no empirical evidence of its active use nor success in defending against stolen credentials or cookie hijacking.

The remainder of this paper is organized as follows. We give an overview of the state of the art and its limitations in Section 2. We introduce our dataset in Section 3. We analyse our results on browser fingerprinting adoption in Section 4. We define our attack models and evaluate them in Section 5. We discuss our results and its limitations in Section 6, before concluding in Section 7.

[5] `https://zenodo.org/record/3872144`

2 Background & Related Work

2.1 Browser Fingerprinting

Browser fingerprinting is a technique to identify a user by leveraging the unique combination of software configurations (*e.g.*, browser, operating system) and hardware characteristics of their device. It was first mentioned as a potential identification technique and studied in 2010 [7]. It combines a set of discriminating attributes mostly accessible from HTTP headers and JavaScript [17]. Commonly accessed attributes are the navigator and screen properties [7, 18], font enumeration [22], canvas [20], audio [8], and WebGL rendering [5].

Studies. Most of the literature has focused on studying the uniqueness and stability of browser fingerprints. They report that browser fingerprinting is a powerful and stateless identification technique [7, 18, 9] to track users for potentially long periods of time [32]. Other studies aimed at estimating the adoption of browser fingerprinting by websites for tracking purposes. They highlighted that 3–5% of the Top Alexa 1M [22, 1, 8] embed browser fingerprinting scripts. However, all these studies only crawled home or random pages, thus lacking deeper insight on more sensitive pages, such as sign-up or payment pages, which demand higher security, but may require more complex user interactions to reach.

Security usages. Bursztein *et al.* [4] argue that canvas fingerprinting can distinguish different families of browsers and uncover the use of PhantomJS to attack a website. Jonker *et al.* [14] studied the "fingerprintability" surface of bots, revealing discriminating attributes to protect websites against web scraping. They also observed that such bot fingerprinting attributes were collected by 15% of the Top Alexa 1M. Vastel *et al.* [33] mention that this technique is already used by websites to block bots. About multifactor authentication, several approaches focused on increasing web session security with browser fingerprinting [29, 2, 27, 15]. However, these studies only cover the methodology and implementation steps, but fail to evaluate their effectiveness in production.

2.2 Multi-factor authentication and session hijacking

New authentication factors are regularly proposed to secure user accounts [25]. However, studies show their adoption is slow [3], leading to compromised accounts if attacks targeting passwords are successful. Sivakorn *et al.* [28] uncovered another attack on user accounts by using cookie session hijacking to log into accounts and steal user sensitive data.

Synthesis. To the best of our knowledge, the state of the art stops either at studying browser fingerprinting from a tracking perspective or for bot detection purposes. It does not deliver any contribution to the adoptions and usages of browser fingerprinting on more sensitive pages, dealing with personal or payment data. The following sections, therefore, propose to address these limits by delivering a new study focusing on the adoption of browser fingerprinting to increase

web security. Our contribution advances the state of the art by considering a dataset of 1,485 real-world sensitive pages collected from 446 domains. Unlike previous studies, we obtain this dataset by manually performing an in-depth exploration of a carefully selected set of domains, thus going beyond the surface of websites to study these sensitive web pages.

3 A Dataset of Secure Web Pages

This section reports on our methodology to build a dataset of secured web pages to study the use of browser fingerprinting for security purposes.

3.1 Websites Under Study

Secured pages. All pages of a website are not equal when it comes to user security. While most web pages do not process sensitive data, some require careful design to deal with user personal information (*e.g.,* emails, credentials, personal details, payment card numbers). On sensitive web pages, any security breach can quickly lead to privacy leaks for the end-users and seriously affect the reputation of the website. We decided to focus on 4 types of web pages requiring personal information or requesting personal data:

1. Sign-up, which may require email, name, password, and additional personal information depending on the website.
2. Sign-in usually requests user credentials (email/pseudonym and password).
3. Payment is a page containing a specific form requesting the user to input their payment information (*e.g.,* credit card, wallet, banking information).
4. Basket refers to any page related to a shopping basket or shopping cart process, starting from adding an item up to, but not including, payment. Such pages may also request additional information, such as billing/delivery addresses.

We call these 4 types of web page *secured web pages*. To assess our results, we also collected samples from other types of pages. From these, we isolated *home pages* as it has been reported they might fingerprint 25% less [30]. We consider pages that are neither secured nor home pages to be *content* pages.

Website categories. Previous studies crawled the Top Alexa with automated tools, thus studying a large set of homepages and resources reachable by bots. We decided to avoid the bias introduced by bots, preferring to manually browse the websites and reach deeper pages that require user interaction. Moreover, we are interested in studying the adoption of browser fingerprinting on secured pages. In this context, the diversity of websites indexed by the Top Alexa —or other ranking lists— proved to be unsatisfactory. Thus, we decided to consider a list of website categories that we estimate to be more relevant for the purpose of our study. To build this list of relevant categories, we adopted the following methodology:

– We targeted websites focused on gambling, credit card, financial and money services.
– We focused on different retail websites, such as event tickets, games, flights and transports, and accommodation booking websites.
– Finally, we added to our list job search, social network, adult, dating, institutional and governmental websites as they often request detailed personal information when creating an account.

The complete list of keywords we used to reach the websites is available in Appendix A. We mainly entered a combination of country name, category and the word '*website*' into the Google search engine, and visited the websites given on the first page of the results. We also translated the search terms into the main language of the country when we were not getting suitable results according to the country and the category, as for example, was the case for Russian websites.

3.2 Web Page Acquisition

Past studies used automated crawls to observe browser fingerprinting at scale [22, 1, 8]. However, relying on bot crawls introduces bias in the collected data [14, 11, 34] as more and more websites use defenses to block bot access [33]. Automating the registration and payment processes is also a challenge because of the high variability that can be found in related forms [13]. No unique or universal standard exists and the number of required fields can strongly differ. The coding practices may be different with obfuscated code and custom attributes, making it hard for a bot to automatically match a field with the right information. Security requirements are also different, including diverse password constraints and security questions. As the scope of this paper is not to develop a bot to automatically test these websites, we manually visited them and collected the required data via a custom web extension we developed. This strategy allows us to appropriately locate interesting secured pages and reduces the bias of being blocked by the security mechanisms in place.

3.3 Fingerprinted Page Attributes

This paper does not intend to discover new browser fingerprinting techniques, but rather to investigate the adoption of existing ones in the context of web security. As part of our data acquisition campaign, we thus focused on collecting the values of all existing attributes reported in the literature. Thus, we consider navigator & screen properties [7, 18, 22, 21, 24], fonts enumeration via `span`'s width and height measurement [22], canvas [20], audio [8] and WebGL rendering [5] and parameters [18], WebRTC [8] and bots detection attributes, including `window` properties that were considered by Jonker *et al.* [14]. Our web extension monitors access to the attributes by overriding getters of selected properties and functions. Whenever one of these attributes is accessed, the web extension collects the function or property name, the list of arguments passed if it is a function, the property's value or the function's return value, the script accessing the property or function, and the page's URL.

We manually visited the selected websites and we used a single identity we created on a popular email provider. For each visited page, we stayed at least 10 seconds, and manually filled each form. When asked for proof of identity, such as a valid phone number, an ID or a credit card, we provided one of the phone numbers used to create the email account. As our identity was fake, we were not able to provide a real ID (*e.g.,* a passport) when required by some websites. Given that payment pages require filling out credit card information, we used a fake credit card generator[6] to be able to validate online payment forms and make sure that we trigger most of the scripts embedded in the page. Even though the generated cards were fake and the payment processes were not completed, we bypassed many client-side verifications thanks to this technique. Yet, using fake payment data raises several issues, such as, our account could be considered suspicious and prevented from performing additional actions, and our IP address could be blacklisted and blocked for the rest of our experiment. Finally, to reduce suspicion, we used several residential IP addresses during our data collection. Although we provided websites with fake payment data, we believe the low number of payment attempts we performed on each website has had minimal impact on their operations. We did not try to harm them in any way and we canceled our baskets if any information received after the payment attempt indicated the website could validate the basket and ask for a future payment.

3.4 Resulting Dataset Description

We performed our data collection campaign from December 2019 to January 2020. We used a fresh install of Chrome 79 on Ubuntu 19.04. We always accepted the default cookie settings for pop-ups, but refused all other types of solicitations, such as geolocation, notifications, or newsletters. In total, we visited 1,485 pages across 446 websites.

Website category and ranking. We used the *category* keyword we put into the search engine to categorize the website. We specifically targeted bank and money-related services because of the sensitivity of the data they manipulate, visiting 85 of these websites (see Figure 1). The country tag represents the main country the website operates in. We assign the country tag by following the result of two observations:

1. Is the website available in English or in multiple languages and translated into the user's preferred language?
2. Are the services proposed by the website available in a single country or geographic zone?

If the website is available in multiple languages or served in English, and if the website provides services to multiple countries, we use the *International* tag. Otherwise, we specify the country. If the website does not operate in a listed country, we use the *Other* tag. With these rules, we tagged 142 *International*

[6] `https://www.creditcardvalidator.org/`

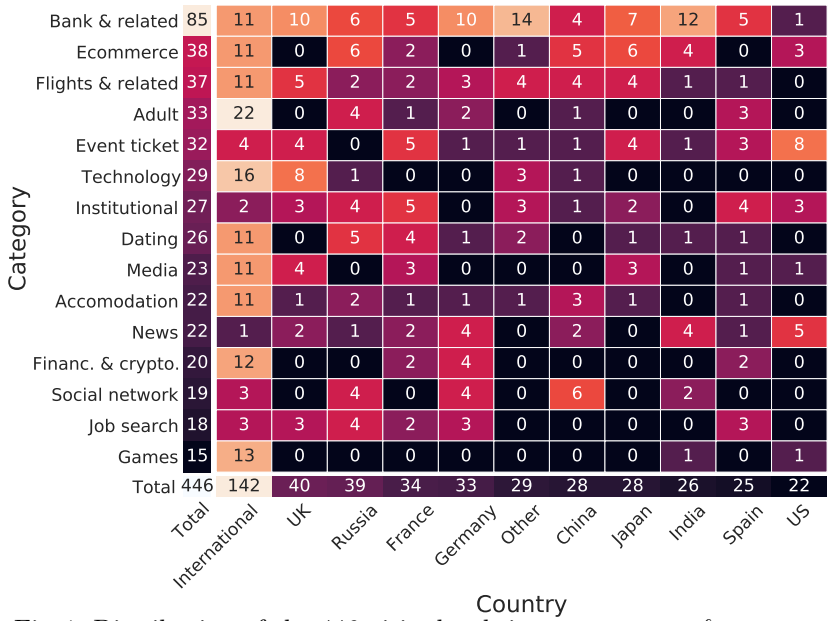

Fig. 1: Distribution of the 446 visited websites per country & category.

websites. The resulting distribution of visited websites per country and category is depicted in Figure 1.

We did not aim to build an exhaustive manual dataset. However, we checked the Top Alexa rankings of the websites in our dataset. We find that our dataset is relatively well balanced across the less-than-1k (18%), 1k-10k (29%), 10k-100k (27%) and higher-than-100k (26%) Top Alexa rankings.

Page type. We also tagged each page according to its type. By default, a page is associated to a single tag, with the exception of *home* or *content* pages that have a sign-up or sign-in embedded form that allows creating an account or authenticating without going to a specific page (44 occurrences in our dataset), and single pages that handle both account creation and authentication processes (3 occurrences in our dataset). In the case of pages containing both basket-like content and a payment form, we tagged the page as payment. If no tag matched, we used the *content* tag. Our dataset is well-balanced between secure (44%) and non-secure (56%) pages. The basket, sign-up and sign-in pages are equally present ($12 - 13\%$) while payment pages represent 6% of our dataset.

Script classification. Several studies exist to classify a real-world dataset of scripts into *fingerprinters* and *non-fingerprinters* [12, 26]. All rely on both automatic and manual classification to combine efficiency and reliability. However, none of these studies provide their implementation, making them difficult to be reused. We designed and implemented an algorithm to classify the scripts of our dataset. We have made its implementation freely available [6].

The algorithm relies on a incremental process to build similarities between scripts based on the attributes they access and proposes a manual step to reinforce the fingerprinting process. When the algorithm is unable to reach a decision on a script, the user can step in to guide the algorithm and provide the correct label. Out of 4,665 scripts, the algorithm provided a label to 4,296 scripts. We manually analyzed, over several iterations, 359 of them. Overall, this classification process found 169 browser fingerprinting scripts included in 405 web pages of our dataset. The information concerning the fingerprinting scripts, their URL and the domains they were found on are available in our public dataset.

The remainder of this paper builds on this dataset of secure pages to identify scripts that collect browser fingerprints and evaluate the additional security layer provided by the browser fingerprinting technique.

4 Analysis of Secure Web Pages

4.1 Browser Fingerprinting Attributes

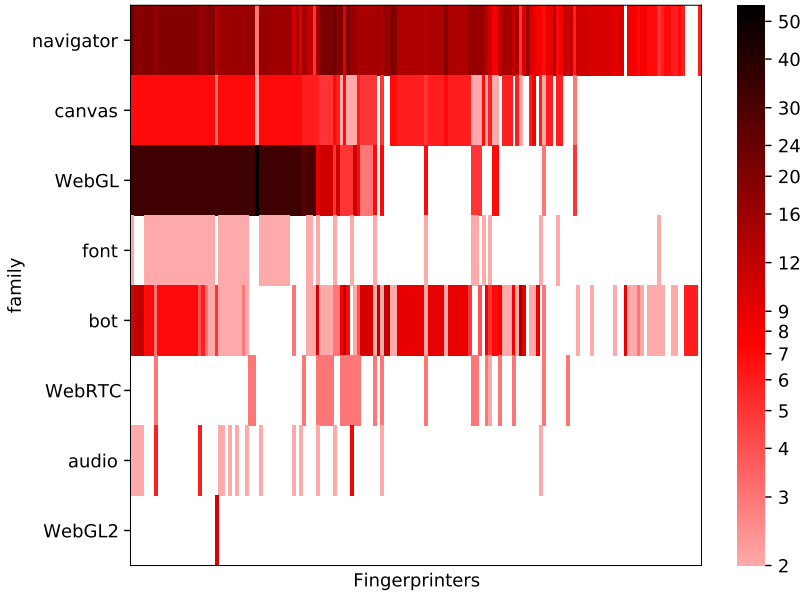

Fig. 2: Distribution of families across fingerprinters.

Of the 169 browser fingerprinting scripts we classified, we observed 132 distinct fingerprinting attributes that we organized into 8 families. The family of an attribute is the parent JavaScript object calling the attribute; except for the bot attributes where we used Jonker *et al.*'s list [14]. Figure 2 reports on the

distribution of the attributes per script grouped by family, showing that all attribute families are exploited in the wild. The most accessed attributes are the `User-Agent`, screen `width` and `height`, `plugins` list, and `timezone`. Even if it would be tempting to rely on these to detect fingerprinting scripts, they can be used for many other purposes, such as analytics or adjusting a website's UI to the device. In our dataset, these attributes are used by 81%, 20%, 19%, 6% and 13% of non-browser fingerprinting scripts, respectively. This illustrates the difficulty of identifying reliable attributes to detect browser fingerprinting scripts.

We analyzed the scripts that use canvas or WebGL fingerprinting. 120 scripts fingerprint browsers using canvas drawing primitives, using between 2 and 14 different drawing instructions. We found 44 different drawing sequences. Concerning WebGL fingerprinting, 54 scripts draw with WebGL primitives, using between 17 and 20 distinct drawing instructions. Only 7 drawing instruction sequences are different. Moreover, one sequence is used by 46 scripts. 54 fingerprinters enumerate fonts. The number of fonts tested ranges from 66 to 594, with 19 different sets of fonts. We observe 2 sets of fonts being largely checked by fingerprinters: a set of 82 fonts tested by 17 scripts, and a set of 66–69 fonts used by 18 scripts. Thus, even though there is a potentially unlimited combination of testable fonts, a majority of scripts use similar sets. We believe this is due to these font sets being copied from one script to another, as well as, being sufficient to capture enough uniqueness. We observed 107 fingerprinters collect at least one bot attribute. The average number of bot attributes is 5. `PhantomJS` attributes are the most collected (41% of all scripts), followed by those that detect `Headless Chrome` (18–33%) and `Selenium` (12–16%).

Finally, we observed that the most used attributes belong to the earliest identified for browser fingerprinting, like the navigator and screen properties (Eckersley *et al.* [7] in 2010) and the canvas (Mowery *et al.* [20] in 2012). More recent attributes are less present in our dataset, such as audio and WebRTC (Englehardt *et al.* [8] in 2016). We also found one fingerprinting script accessing 9 unpublished attributes from the `WebGL2RenderingContext` object which is part of the WebGL2 APIs.

4.2 Origins of Browser Fingerprinting Scripts

Regarding the adoption of browser fingerprinting for security purposes, we analyzed the scripts hosted by domains whose main goal is security. We analyzed their target markets and their presence in our dataset and identified 14 fingerprinting scripts from 12 security-focused organizations. For each of the organizations, we extracted their main purpose, and analyzed the presence of their scripts on the sensitive page types we defined. Table 1 reports on these results. All of these security scripts are present in at least one of our 4 sensitive page types.

We analyzed the attribute families collected by these browser fingerprinting scripts. All major techniques are being actively used. The navigator and screen properties are the most collected (included in 14 scripts), followed by canvas (12), bot (11), WebGL parameters (10), WebGL drawing (6), audio and font

Table 1: Summary of security organizations, with the accessed attributes and the presence in the web pages of our dataset.

Organization goal	Organization name	Script # of attributes	Script presence on					
			# domains	# pages	sign-up	sign-in	basket	payment
Payment platform	Adyen	47	1	1				✓
	CentroBill	14	1	1			✓	
	Probiller	29	1	4			✓	✓
	Razorpay	10	1	1				✓
	Secured Touch	73	1	6		✓		
Fraud prevention	Iovation	8	1	1				✓
	Nudata Security	29	2	3	✓	✓		
	Sift Science	26	10	26	✓	✓	✓	✓
	Simility	49	2	3				✓
Bot protection	Datadome	33	1	1		✓		
	Geetest	64–65	4	7	✓	✓		
	PerimeterX	69	1	3				✓

enumeration via span (5), canvas font enumeration and WebRTC (3). Access to `navigator.userAgent`, `navigator.platform` and `navigator.vendor` was found in 13 scripts. These navigator attributes overlap, we believe that they are used to detect spoofing. Moreover, we observed 13 scripts where `screen.width`, `screen.height`, `screen.availWidth`, and `screen.availHeight` are collected. These attributes can also be used to detect spoofing, as the available sizes should be smaller than the width and height. Jonker shows this invariant can detect bots [14]. The 3 organizations that claim to protect against bots naturally collect bot attributes. PerimeterX collects 10 of them, and Datadome 5, both covering all major bot types. However, Geetest only collects 2 bot attributes, both for detecting PhantomJS.

4.3 Secured vs non-secured web pages

We analysed the ratio of webpage types that include a browser fingerprinting script. We found browser fingerprinting scripts on all types of web pages. Basket (33.8%) and Sign-up (31.1%) pages fingerprint more than the average, followed by *content* (25.6%), Payment (25.3%), Sign-in (23.4%), and *home pages* (23.0%). Other studies have not targeted these specific page types and have generally relied only on home pages. Consequently, we are—to the best of our knowledge—the first study to observe the prevalence of browser fingerprinting in sensitive and secure web pages. We compared fingerprinting in secured to non-secured pages. We found 54 scripts exclusive to secured pages, 68 scripts exclusive to non-secured pages, and 47 in both.

We counted the number of fingerprinting scripts included on a page. Out of 405 pages that fingerprint, 339 pages include 1 script, 51 pages had 2 scripts, and 15 pages had 3 scripts. Out the 66 multi-script pages, (*i.e.*, 51+15), 10 had first-party fingerprinting scripts, 27 served fingerprinting scripts from a different domain, and 29 served fingerprinting scripts from both first and third-party domains. We make several hypotheses based on our observations. First, the browser

fingerprinting scripts have different purposes, such as advertising or security services, and likely do not share the fingerprints they collect. Next, in the case of pages from websites being developed by several teams, they may integrate multiple browser fingerprinting scripts unintentionally. A majority of the pages with multiple fingerprinting scripts are secured pages, (35 secured versus 31 non-secure pages), although secured pages represent only 44.2% of our dataset. This result supports the statement that secured pages fingerprint more aggressively than non-secured pages.

4.4 Additional Security Mechanisms

During our data acquisition, we also marked the usage of any MFA mechanisms or bot detection techniques we found. We observed 38 pages with an MFA mechanism, the majority being sign-up pages. 3 distinct mechanisms were used during our collection: an email code or confirmation (used 19 times), an SMS code (17), and a phone call (2) in which the code to enter on the website consists of the last x digits of the calling number. The usage of an MFA mechanism for sign-up pages implies a stronger requirement for proof of identity. Because browser fingerprinting could fulfil this requirement, we compare the average number of fingerprinting attributes on these pages to other pages. We observe only 1 sign-up page that contains an MFA mechanism also embedded a browser fingerprinting script. Moreover, this webpage also included a bot detection mechanism. This means that they used both an authentication confirmation and a verification for bots. Thus, the presence of a browser fingerprinting script might be used to serve either of these purposes, as we are only observing from the client-side, we cannot conclude.

Regarding bot detection, we found 51 scripts using bot detection mechanisms: 41 ReCaptcha, 6 Geetest puzzles,[7] and 4 textual captchas. As for the MFA mechanisms, they were mainly observed in sign-up pages. 2 pages were using 2 bot detection techniques: a sign-in page on an adult website and a sign-up page on an event ticket website. Half of the pages with a bot detection mechanism embed a browser fingerprinting script. This shows the interest of websites in using bot detection techniques based on browser fingerprinting.

Synthesis. In this section, we explored the browser fingerprinting scripts of our dataset, their presence on the different page types we considered and the adoption of fingerprinting in combination with additional security mechanisms. More specifically, we show that browser fingerprints are effectively accessed for all the types of web pages covered by our study. We finally observed that browsers are fingerprinted slightly more aggressively on secured pages.

[7] `https://www.geetest.com/en/demo`

5 Attack models

This section introduces two attack models we used to assess the security benefits of browser fingerprinting.

5.1 Stolen credentials

Extracting the protected websites We observed in Section 4 that browser fingerprints are collected by websites during the authentication process. We are interested in observing any security improvements brought by fingerprinting in the case of stolen credentials. We assume that a hacker steals a user's credentials—through a data leak, a phishing page, or any other technique—and tries to authenticate into the targeted website. We reproduced this attack behaviour by trying to log into the accounts we created for this experiment. It is worth mentioning that we used a phone number or any additional information needed to create the account, but we skipped anything that was not mandatory. We assume that the attacker may use a different browser instance on a different OS, with different cookies than the victim's browser, while browsing from a different network than the network associated with the original accounts. We assume that the attacker can solve Captchas when using stolen credentials. Therefore, bot detection mechanisms are not a reliable protection in this context. We ran this attack on the 42 websites we were able to create accounts on—12 of them use a browser fingerprinting script on the sign-in page. The 42 websites are well balanced concerning the *Country* tag we defined, and mainly concern cryptocurrencies, money transfer, e-commerce, adult, event and sport tickets content. Among these websites, 16 of them belong to the Top 1k Alexa, 8 between 1k and 10k, 11 between 10k and 100k, and 7 above the Top 100k. We expect websites that collect browser fingerprints to use it them protect the accounts from stolen credentials. Our attempts to log in to the accounts with different fingerprints fell into the following 3 cases:

1. We were able to log into 37 websites without facing additional multi-factor authentication mechanisms or security warnings.
2. Three websites sent a warning message about an unknown connection to our account. These messages contained the IP address, the OS and the browser we tried to connect with.
3. Two websites asked for additional proof of identity. The first one sent an email code with additional information about the ongoing connection. The other sent an SMS code to enter to validate the connection attempt. Those 2 websites also proposed a security panel where the user can check their trusted devices.

We observe that only 5 out of the 42 websites react in a manner that strongly suggests fingerprint-based detection of known devices and browsers is being used to secure the account, namely Google, WeTransfer, (files transfer service) Skrill, Crypto and Binance (cryptocurrencies, finances or money transfer websites). 4 of them have a security panel with the authorized devices with their characteristics and all the connection attempts to log into the account.

Table 2: Parameters and results concerning the reauthentication experiment. *Different IP, browser, device* indicates the IP address, the browser and the device were different from the ground truth respectively. * indicates the cookies from step 5 were reused in step 6 (but they differ from the ground truth cookies).

Website	MFA on sign-up pages	FP on		Authentication attempt combinations					
		sign-up	sign-in	n°1 ground truth	n°2 different IP	n°3 different IP no cookies	n°4 different IP diff browser no cookies	n°5 different IP diff browser diff device no cookies	n°6 same IP diff browser diff device same cookies*
Google	SMS OTP			connection	connection	SMS OTP	SMS OTP	connection + alert	connection + alert
Skrill				connection	connection	SMS OTP	SMS OTP	SMS OTP	SMS OTP
Crypto	SMS OTP	✓	✓	connection + alert	connection + alert	connection + alert	connection + alert	connection + alert	connection + alert
WeTransfer				connection	connection	connection + alert	connection + alert	connection + alert	connection + alert
Binance	Email OTP	✓	✓	connection	connection	connection + specific alert	Email OTP	Email OTP	connection + specific alert

Isolating the triggering characteristics As we noticed during our previous experiment, several details, including the OS, browser and IP address, were provided to the user to explain the warnings or requirements for additional information to authenticate. The IP address can be used to extract the approximate geolocation of the user and to detect connections from unusual networks (*e.g.,* through a cloud provider). Although not indicated, we believe the presence of previous login cookies might also be used by the website to decide to authenticate the users more easily. Their absence might reveal a device or browser change. For the 5 websites of our previous results, we tried to isolate the set of elements that trigger an additional security authentication mechanism or warning. To do so, we tried the following 6 combinations:

1. We re-authenticated with the same conditions as the ones the account was created with, to get the ground-truth.
2. We signed-in with the same browser but using a different IP address than the one used for the account creation and previous login attempt.
3. On a different IP address, using the same browser, we logged in using the browser's private mode to navigate without reusing any cookies.
4. On the same device but using a different browser, with a different IP address.
5. We then tried to reauthenticate with a different device and browser, and a different IP address.
6. Finally, we tried to log again on the same device and the same browser as the previous combination, without deleting cookies or any other stateful data, but we changed our IP address to reuse the original IP address used to create the account and log in for the ground truth combination.

We ran the above combinations in order. Browser and OS changes make the fingerprint different, contrary to IP address and cookies that do not affect the browser fingerprint. If fingerprinting is used to secure the account, we expect to observe different behaviors on the combinations where the fingerprint changes, namely combinations n°4, n°5 and n°6.

Table 2 summarizes up the results we observed according to the changes we applied to the browsing characteristics. Google, Skrill, and WeTransfer seem to be based on cookies. When they are not present, the first 2 ask for a *One Time Password* (OTP), while WeTransfer sends an alert. Crypto always has the same behavior: it allows the connection, but sends an alert. Finally, Binance have the most advanced system. First, it sends an alert about the IP address when we changed it, when browsing without cookies (combination n°3). The message did not contain any reference to device or browser changes, so we believed the website knew we were on the same device and browser. The behavior changed when we used another browser on the same device: Binance sent us an email containing an OTP and basic information about the new browser used (combination n°4). It also sent an email with an OTP when using a different device (combination n°5). When staying on the browser chosen on this second device and using the same IP address as the one used, we received an alert (combination n°6). Based on this last experiment, we make 2 observations:

- The alert message is different from the one when we changed the IP address. The message now mentions a change in the browser and device.
- We received an email alert, but we were still able to sign in. We believe this is because the cookies set by the browser were the same as the ones in the previous combination when we needed to provide an email OTP to validate the connection.

In our dataset, we see browser fingerprinting being used only once, in combination with other identifying techniques, to resist stolen credentials. As this attack is similar to a user trying to login from a fresh browser, it also illustrates the additional steps users needs to complete to sign in with a new browser.

5.2 Cookie hijacking

Attack design Cookie hijacking can lead to account compromise and data leaks [28]. As browser fingerprinting can be used to identify a browser, we make the hypothesis it can be used to help tell if a cookie has been hijacked and used by a different browser. Our goal is not to study the existing ways to perform a cookie hijacking. In our attacks, we assume an attacker was able to steal cookies, no matter the method used—XSS vulnerability, insecure network exchanges, malicious JavaScript injection. Instead, we aim at studying the resilience to cookie hijacking by websites in our dataset if browser fingerprinting is used to protect the accounts. We designed 2 attacks to study cookie hijacking.

Our first attack is session cookie hijacking. It consists in trying to authenticate using cookies stolen from an existing user session. We log in to the target site on a first browser, then we extract the cookies and login page URL and insert these into a second browser. If the attack works, the second browser will be authenticated and the session will be in the same state as on the first browser. If not, the second browser will be stuck on the login page.

Our second cookie hijacking attack focuses on basket workflows. The goal is to obtain the same basket as a user by hijacking their cookies. We fill a basket with a commercial item, and visit the page summarizing the basket and its

content. Similarly to the session hijacking cookie attack, we then extract the URL and cookie, and put them in another browser. If the basket content is the same on the 2 browsers, the attack is successful.

Methodology For each website, we automated the browsing to the required pages with a Puppeteer instance. We automate the insertion of cookies and the navigation to the URL with a second Puppeteer instance. We lower the possibility to be detected as a bot by changing the fingerprint of the Puppeteer instances for them to look like Chrome 84. To do so, we reused the value of each attribute collected by fingerprinters during our manual data collection and integrated them into an extension in the Puppeteer instance that returns the corresponding value when an attribute is accessed. We also added a delay of at least one second between each action on a page.

Before studying the impact of fingerprint modification, we performed a preliminary run with and without the collected cookies to make sure that sessions could be stolen from the Puppeteer instance and that no other parameters, like `localStorage` or a hidden parameter in the URL, would impact our measurement. This way, we created a subset of websites where our attacks are successful. Finally, we ran our attack on all the websites of this subset and collected the cookies and URLs. We used different parameters and configurations of the second Puppeteer instances by running them on a different device on a different network with a different IP address. We also changed all the fingerprint attributes we monitored during our data collection by giving them values from a Firefox 72 instance with the same extension as described earlier in this section. Should a website be protected and detect the different fingerprint, we rerun the attack by modifying parts of the fingerprint to detect which attributes or combinations trigger the defense mechanism.

Results We ran these experiments in July and August 2020. We used the 42 websites we were previously able to create an account on for the session hijacking cookie. Concerning the basket hijacking cookie attack, we used the 84 websites of our dataset containing at least a basket page—33 of them contain at least one fingerprinter on a basket page. We then ran each step of our validation process to make sure the cookies were the only variable needed to retrieve the basket or session state. The results are presented in Table 3. Because of the time gap between this experiment and the one described in Section 4, we were unable to fill baskets for several websites with

Table 3: Number of websites involved in each step of the validation for the session and basket cookies attacks, and results of the attack on the validated subset

	# websites			
	Session		Basket	
	FP	no FP	FP	no FP
Dataset	12	30	33	51
Cannot automate	0	5	8	8
Anti-bot triggered	2	1	3	3
Impacted by other params	3	5	6	6
Nothing sold	0	0	7	16
Validated subset	**7**	**19**	**9**	**18**
Attack works	**7**	**19**	**9**	**18**

a single item as some of them were not selling anything anymore. We believe this is likely due to the economic and societal restrictions following the Covid-19 pandemic. We ended up with a validated subset of respectively 26 and 27 websites for our session and basket cookies attacks. As explained in the methodology section, we then ran our attack and inserted the cookies on a different device on a different network with a different fingerprint and HTTP headers. With these parameters, the attacks worked on every website of our validated subset. These behaviors imply no defense mechanism was being used. Thus, browser fingerprinting is not used to protect against a session or basket cookie hijacking on the websites of our dataset.

As we did not detect any usage of additional security mechanisms, we studied the way *HTTPS* and *HSTS* are deployed and how cookies are configured to observe if their settings were secure enough to protect against traffic sniffing. If these elements are properly set, it lowers the attack surface on cookies by complicating their extraction via JavaScript and avoiding their theft from HTTP requests [28]. Over the 53 websites we tested our cookie hijacking attacks on, 52 were redirecting their traffic through *HTTPS* and 30 of these 52 websites were setting the *Strict Transport Security* response header in the browser. During the experiments, we collected and injected $1,080$ cookies, 198 (18%) and 305 (28%) were `HTTPOnly` and `Secure`, respectively. We also looked at the `SameSite` parameter, observing 11 (1%) and 109 (10%) cookies have a `Strict` and `Laxist` `SameSite` policy, respectively. Even if the `SameSite` parameter is now set by default to `Laxist` since Chrome 80/Firefox 69, few websites were setting it to a secure value, indicating they were added before to all requests because of the default `None SameSite` policy.

Based on these observations, we conclude that developers put a lot of trust in cookies as their presence alone in our tests lead to direct user authentication. This trust is only possible thanks to strong security mechanisms in browsers that have grown and matured a lot in the past decade. The rise of *HTTPS* coupled with a lot of control over what can be executed on a webpage (through *CSP*, *CORS* and all their derivatives) have changed the way we come to reason about cookie hijacking and how much harder it is to pull off such an attack today. Yet, our experiment shows that if indeed cookies are stolen, none of the tested websites have any mechanisms in place to detect any irregularities. We can only hypothesize at this point that this may not be in the scope of their threat model.

Synthesis. In this section, we designed 2 attack models and tested them to measure the effectiveness of fingerprinting to protect users on web pages in our dataset. We observed fingerprinting being successfully used to improve security in our first attack using credentials to log in. Concerning our second attack, we did not detect any website in our dataset that used browser fingerprinting to protect against cookie hijacking.

6 Discussion

Understanding the intent of a fingerprinting script is hard. In the case of browser fingerprinting, analyzing why a script is included in a web page and why it's accessing specific attributes is complex as there is little indication of what will be done with the collected data once it has left the device. Still, it is possible to rely on some signs to capture the intent behind a fingerprinting script, such as:

- **Accessed APIs**: depending on the goal of the script, some APIs may be picked over others. For example, anti-bot companies access well-identified bot attributes, while others interested in cross-browser fingerprinting access OS and hardware-specific attributes.
- **Number of collected attributes**: while a very high number of attributes can often be linked to a fingerprinting behavior, the numbers vary. As seen in Table 1, some third parties, like Iovation, build on only 8 attributes, while others, like Secured Touch, collect up to 73. As a lot of the state of the art in fingerprinting is interested in either uniquely identifying devices or detecting inconsistencies, it makes sense to collect as many attributes as possible. Yet, as seen with Iovation, if you have a clear goal in mind, collecting very few attributes can be enough for your purpose.
- **Execution context**: where the fingerprinting script is located can show intent. If a fingerprinting script is included in all web pages, it is probably linked with an anti-bot system but, if it is only present on a payment page, then it is likely used for fraud prevention.

Considering the above signals, it is possible to estimate how the collected information will be used, but it does not provide certainty without having access to the backend where the browser fingerprints are analyzed.

Fingerprinting is not being used to protect individual accounts. In Section 4.2, we identified third-party actors who collect a wide range of data to implement bot protection and fraud prevention. They protect a website globally against external threats. Yet, when looking at what is offered to protect users' accounts, the story seems very different. Based on our experiments detailed in Section 5, there is little evidence that fingerprinting is currently being used to protect individual accounts. As we detected fingerprinting scripts delivered by 12 security-oriented organizations, we would have expected them to add an additional security layer to protect users. This raises the question of the relevance of using such a script from a security organization if the final usage is not security. More generally, we tested the defenses of 42 websites by creating accounts and logging with several contexts and parameters. Apart from some warning messages with few details on the new connecting device, we found only a single website blocked access to their services when the browser fingerprint did not match. Moreover, we have not detected any usage of browser fingerprinting to protect against our second attack, the cookie hijacking. We believe these are negative results of our paper and deserves further discussion.

First, theses results raise the question of why we observed such behaviors. One concern could be the accuracy of the browser fingerprinting algorithm. While cookies and IP addresses send strong signals that websites have relied on for years, a fingerprint is, in contrast, more volatile. It can change due to a minor modification to the browser's configuration or an update. Some attributes may be deemed too unstable to be included for verification while others are much more reliable and even predictable. As detailed by Vastel *et al.* [32], browser fingerprinting techniques require constant adaptation to maintain their robustness. Another concern is user experience, as having an overly sensitive algorithm could prompt for additional checks too often, even if the user did not change their device or browser.

Deficiencies in the state of the art. As we identify concerns about the use of browser fingerprinting in a multifactor authentication system, we believe the state of the art currently lacks studies to measure the effectiveness and reliability of MFA with browser fingerprinting. First, users would need a way to add a new fingerprint to their account to be able to connect with another device. Websites in our dataset seem to use a OTP email. We believe other options should be studied because each authentication system is different and has its own trade-offs. Also, fingerprints evolve over time, and a multifactor authentication system would must be able to tell if a fingerprint is an evolution of an already registered one or not. While solutions have been proposed to compute a fingerprint evolution [32], it has been shown to not be fast enough when confronted with a large dataset [19]. Used in an authentication context, it would require a quick decision to have negligible impact on the user experience. An interesting study has been proposed by Alaca *et al.* [2] about the requirements of a such a system, but due to the rapid evolution of the web ecosystem, the study might be outdated. Finally, the state of the art lacks an evaluation of the user experience, satisfaction and confidence when using this kind of system.

Benefits provided by our dataset. We believe our manual dataset is interesting for the community for future research on browser fingerprinting. In combination with a study about the origin and causality of a change in the browser fingerprint, it could be used to better understand what information the websites is looking for. They could be interested in an attribute that concerns some specific hardware or the software detail of the device, for example, to determine whether the fingerprint is consistent. An inconsistency could reveal a possible identity theft or a fraudster. Thus, it could help understand the purpose of browser fingerprinting collection on sensitive pages. Moreover, the automated detection of multi-factor authentication mechanisms suffers from the same biases as the automated detection of browser fingerprinting, and much of the same reasoning behind this study would apply. As explained in Section 2, the literature lacks a study on multi-factor authentication adoption on the web. Our dataset provides information about such mechanisms and could be used as a starting point for an in-depth study on these security systems.

7 Conclusion

In this paper, we studied the adoption of browser fingerprinting for security applications. More specifically, we analyzed 4 types of secured web pages—sign-up, sign-in, basket, and payment—that process sensitive personal data. We considered the state-of-the-art JavaScript attributes and developed an extension to monitor browser fingerprinting attribute accesses. To avoid biases introduced by automated crawlers and bots, we manually visited 1,485 pages published by 446 websites, and created accounts, logged in to verify authentication procedures, and went through the payment processes where available. We labeled 169 distinct fingerprinters using an existing technique. We publicly share our secured web page dataset and the detected fingerprinters we found.[8] We observed these fingerprinters being served by all types of secured pages and various website categories. We analyzed the script providers and found 12 security-focused organizations that use browser fingerprinting in secured web pages. We measured the use of MFA systems and bot detection, showing fingerprinting is used in combination with other identification techniques. We defined 2 attack models, stolen credentials and cookies hijacking, and evaluate websites in our dataset against them. Finally, we did not observe fingerprinting being actively used to secure websites against these 2 attacks.

References

1. Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., Diaz, C.: The web never forgets: Persistent tracking mechanisms in the wild. CCS'14
2. Alaca, F., van Oorschot, P.C.: Device fingerprinting for augmenting web authentication: Classification and analysis of methods. ACSAC'16
3. Bursztein, E.: The bleak picture of two-factor authentication adoption in the wild (2018), https://elie.net/blog/security/the-bleak-picture-of-two-factor-authentication-adoption-in-the-wild/
4. Bursztein, E., Malyshev, A., Pietraszek, T., Thomas, K.: Picasso: Lightweight device class fingerprinting for web clients. SPSM '16
5. Cao, Y., Li, S., Wijmans, E.: (cross-)browser fingerprinting via os and hardware level features. NDSS'17
6. Durey, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: An iterative technique to identify browser fingerprinting scripts (2021)
7. Eckersley, P.: How unique is your web browser? PETS'10
8. Englehardt, S., Narayanan, A.: Online tracking: A 1-million-site measurement and analysis. CCS'16
9. Gómez-Boix, A., Laperdrix, P., Baudry, B.: Hiding in the crowd: An analysis of the effectiveness of browser fingerprinting at large scale. WWW'18
10. Group, A.P.W.: Phishing activity trends report. https://docs.apwg.org/reports/apwg_trends_report_q3_2019.pdf (2019)
11. Invernizzi, L., Thomas, K., Kapravelos, A., Comanescu, O., Picod, J., Bursztein, E.: Cloak of visibility: Detecting when machines browse a different web. S&P'16

[8] https://zenodo.org/record/3872144

12. Iqbal, U., Englehardt, S., Shafiq, Z.: Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors
13. Jonker, H., Kalkman, J., Krumnow, B., Sleegers, M., Verresen, A.: Shepherd: Enabling automatic and large-scale login security studies (2018)
14. Jonker, H., Krumnow, B., Vlot, G.: Fingerprint Surface-Based Detection of Web Bot Detectors (2019)
15. Laperdrix, P., Avoine, G., Baudry, B., Nikiforakis, N.: Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. DIMVA'19
16. Laperdrix, P., Baudry, B., Mishra, V.: Fprandom: Randomizing core browser objects to break advanced device fingerprinting techniques. ESSoS'17
17. Laperdrix, P., Bielova, N., Baudry, B., Avoine, G.: Browser fingerprinting: A survey. TWEB'20
18. Laperdrix, P., Rudametkin, W., Baudry, B.: Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. S&P'16
19. Li, S., Cao, Y.: Who touched my browser fingerprint?: A large-scale measurement study and classification of fingerprint dynamics (2020)
20. Mowery, K., Shacham, H.: Pixel perfect: Fingerprinting canvas in HTML5. W2SP'12
21. Mulazzani, M., Reschl, P., Huber, M., Leithner, M., Schrittwieser, S., Weippl, E., Wien, F.: Fast and reliable browser identification with javascript engine fingerprinting. W2SP'13
22. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. S&P'13
23. Nikiforakis, N., Joosen, W., Livshits, B.: Privaricator: Deceiving fingerprinters with little white lies. WWW'15
24. Olejnik, Ł., Acar, G., Castelluccia, C., Diaz, C.: The leaking battery. In: Data Privacy Management, and Security Assurance (2016)
25. Ometov, A., Bezzateev, S.V., Mäkitalo, N., Andreev, S., Mikkonen, T., Koucheryavy, Y.: Multi-factor authentication: A survey. Cryptography (2018)
26. Rizzo, V., Traverso, S., Mellia, M.: Unveiling web fingerprinting in the wild via code mining and machine learning. PETS'21
27. Rochet, F., Efthymiadis, K., Koeune, F.A., Pereira, O.: Swat: Seamless web authentication technology. Association for Computing Machinery (2019)
28. Sivakorn, S., Polakis, I., Keromytis, A.D.: The cracked cookie jar: Http cookie hijacking and the exposure of private information. S&P'2016
29. Unger, T., Mulazzani, M., Frühwirt, D., Huber, M., Schrittwieser, S., Weippl, E.: Shpf: Enhancing http(s) session security with browser fingerprinting (AReS'13)
30. Urban, T., Degeling, M., Holz, T., Pohlmann, N.: Beyond the front page: Measuring third party dynamics in the field
31. Vastel, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: Fp-scanner: The privacy implications of browser fingerprint inconsistencies. USENIX'18
32. Vastel, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: FP-STALKER: Tracking Browser Fingerprint Evolutions. S&P'18
33. Vastel, A., Rudametkin, W., Rouvoy, R., Blanc, X.: FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. MADWeb'20
34. Zeber, D., Bird, S., Oliveira, C., Rudametkin, W., Segall, I., Wollsen, F., Lopatka, M.: The representativeness of automated Web crawls as a surrogatefor human browsing. WWW'20

A Selected Search Keywords

We used the following list of keywords to get specific website types: – Bank – Money transfer service – Stock trading – Financial – Cryptocurrency – Social insurance – Taxes – Healthcare – Job search – News – Email – Adult – Dating – Metro/train/flight tickets – Flight companies – Travel agencies – Airlines – Event ticket – Sport ticket – Social network – Ecommerce – Shopping – TV channel – Streaming – Bet games – Poker – Online game.

We used the following list of countries for our experiment: – United States – Japan – Germany – France – Russia – Spain – United Kingdom – India – China